2016 Spring CS300 Homework #1

(Due at Mar. 21 AM 10:30 on classroom)

1. Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in A[1]. Then find the second smallest element of A, and exchange it with A[2]. Continue in this manner for the first $n-1$ elements of A.

   1-1. Write pseudocode for this algorithm, which is known as selection sort.

   1-2. What loop invariant does this algorithm maintain?

   1-3. Why does it need to run for only the first $n-1$ elements, rather than for all n elements?

   1-4. Give the best-case and worst-case running times of selection sort in Θ-notation.

2. Although merge sort runs in $\Theta(n\,lg\,n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to coarsen the leaves of the recursion by using insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

   2-1. Show that insertion sort can sort the n/k sublists, each of length k, in $\Theta(nk)$ worst-case time.

   2-2. Show how to merge the sublists in $\Theta(n\lg(n/k))$ worst-case time.

   2-3. Given that the modified algorithm runs in $\Theta(nk + n\lg(n/k))$ worst-case time, what is the largest value of k as a function of n for which the modified algorithm has the same running time as standard merge sort, in terms of Θ-notation?

   2-4. How should we choose k in practice?

3. Rank the following functions by order of growth; that is, find an arrangement of $g_1, g_2, \ldots, g_{20}$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{19} = \Omega(g_{20})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

$lg(lg^* n)$	$n\,ln\,n$	$n!$	$10^n + n^{20}$	$\sqrt{n}$
$(lg\,n)^2$	$2^{n!}$	4^n	$n^n + ln\,n$	$lg(n!)$
$(lg\,n)!$	$ln\,ln\,n$	$n^{2+sin\,n}$	$n^{1/lg\,n}$	e^n
$5n^2 + 7n$	$n^{5/2}$	$8n + 12$	$5^{lg\,n}$	n^n

4. Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

   4-1. $f(n) = O(g(n))$ implies $lg\big(f(n)\big) = O(lg\big(g(n)\big))$, where $lg\big(g(n)\big) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n

   4-2. $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$

   4-3. $f(n) = O(\big(f(n)\big)^2)$

   4-4. $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$

   4-5. $f(n) + o\big(f(n)\big) = \Theta(f(n))$

5. Solve the following recurrences by giving tight Θ-notation bounds. Justify your answers.

   5-1. $T(n) = 2T(n/3) + n\,lg\,n$

   5-2. $T(n) = T(n-2) + lg\,n$

   5-3. $T(n) = T(n/2) + T(\sqrt{n}) + n$